Mühendislik Bilimleri ve Tasarım Dergisi
6(3), 532 – 540, 2018
e-ISSN: 1308-6693

Araştırma Makalesi

Journal of Engineering Sciences and Design
DOI: 10.21923/jesd.442915

Research Article

KONTROL ALAN AĞLARI İÇİN OPTİMUM STATİK MESAJ ZAMANLAMASI

Esin YAVUZ[*], Sertaç Selim SARICA[2], Ekrem ARTUÇ[3]

[1] Süleyman Demirel Üniversitesi, Mühendislik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü, Isparta, Türkiye
[2] Süleyman Demirel Üniversitesi, Bilgi İşlem Daire Başkanlığı, Isparta, Türkiye
[3] Süleyman Demirel Üniversitesi, Fen Edebiyat Fakültesi, Fizik Bölümü, Isparta, Türkiye

Anahtar Kelimeler
CAN,
Statik Zamanlama,
RM algoritması,
SocketCAN.

Öz

Kontrol Alan Ağı (CAN), yüksek performanslı ve yüksek güvenilirliği olan gerçek zamanlı bir haberleşme protokolüdür. Bu çalışmada, SocketCAN ile bir CAN ağı tasarlanmış ve RM algoritması kullanılarak optimum statik mesaj zamanlaması gerçekleştirilmiştir. SocketCAN, Linux işletim sistemi için geliştirilen bir CAN uygulamasıdır. CAN ağının performansı, öncelik karar mekanizmasına göre belirlenen zamanlamaya bağlıdır. Optimum iletim için anahtar faktör, mesaj önceliklerinin belirlenmesidir. RM zamanlama politikasına göre, iletilecek olan mesajların öncelikleri iletim başlamadan önce belirlenir ve bir mesajın son teslim süresi (deadline), periyoduna eşittir. En erken son teslim süresi olan mesaj, en yüksek önceliği alır.

OPTIMUM STATIC MESSAGE SCHEDULING FOR CONTROLLER AREA NETWORKS

Keywords
CAN,
Static Scheduling,
RM algorithm,
SocketCAN.

Abstract

Controller Area Network (CAN) is a real-time communication protocol with high performance and high reliability. In this paper, a CAN network was designed with the SocketCAN and optimum static message scheduling was realized with using the RM algorithm. SocketCAN is a CAN implementation developed for the Linux operating system. The performance of the CAN network depends on the scheduling determined by the priority decision mechanism. The key factor for optimum message transmission is to determine message priorities. According to the RM scheduling policy, the priorities of the messages are determined before the transmission starts and deadline of a message is equal to its period. The message with the earliest deadline has the higher the priority.

Alıntı / Cite

Yavuz, E. Sarıca. S.S., Artuç, E., (2018). Kontrol Alan Ağları için Optimum Statik Mesaj Zamanlaması, *Journal of Engineering Sciences and Design*, 6(3), 532 – 540.

Yazar Kimliği / Author ID (ORCID Number)
E. Yavuz, 0000-0002-8077-5353
S. S. Sarıca, 0000-0003-0128-0238
E. Artuç, 0000-0001-7452-6249

Makale Süreci / Article Process	
Başvuru Tarihi /Submission Date	12.07.2018
Revizyon Tarihi / Revision Date	11.09.2018
Kabul Tarihi / Accepted Date	24.09.2018
Yayım Tarihi / Published Date	27.09.2018

1. Giriş

Gerçek zamanlı haberleşme sistemlerinde, gönderilecek mesajların son teslim süresi içinde ve tam olarak iletilmesi gerekir. Gerçek zamanlı haberleşme sistemlerinde meydana gelebilecek bilgi kayıpları ya da bilgi akışındaki kesilmeler hizmet

[*] İlgili yazar / Corresponding author: esinyavuz@sdu.edu.tr, +90-246-211-1370

kalitesini önemli ölçüde azaltır. Sistemin sadece çok yüksek hız performansı göstermesi değil, zamanlama performansının da kararlılık göstermesi gerekir. Gerçek zamanlı bir haberleşme sisteminde, sistemin istenen ve öngörülebilir bir şekilde davranmasını sağlamak için uygun zamanlama algoritması kullanılmalıdır. Zamanlama algoritmaları, isteklere hızlı bir şekilde cevap vererek, kullanıcının beklentilerini karşılamalıdır. Aynı zamanda, zamanlama ile mesajların son teslim süresine riayet edilmeli, veri kaybı olmamalı, sürecin sunumu ve sonlanması arasındaki süre minimum iken, üretilen iş maksimum olmalıdır.

Gerçek zamanlı haberleşme sistemleri, Katı Gerçek-Zamanlı Sistemler ve Esnek Gerçek-Zamanlı Sistemler olarak sınıflandırılırlar. Katı Gerçek-Zamanlı sistemlerde zaman kısıtlaması vardır ve zaman sınırı kesinlikle aşılmamalıdır. Bu zamanlama politikaları, geçici ya da kalıcı arızalar olsa bile, her koşulda görevlerin son teslim süreleri içinde iletileceğinin garantisini vermelidir. Zaman gereksinimleri ve arıza modeli, uygulamaya ve ortamına bağlıdır (Oliveira, 2003). Herhangi bir hata nedeni ile geciken veri kullanışsız bir veridir ve tüm sistemin kullanım dışı kalmasına sebep olarak felakete yol açabilir (Rouhifar ve Ravanmehr, 2015). Katı gerçek zamanlı sistemlerde, kritik işlemlerin zamanında yürütülmesini sağlamak için, tüm olaylar önceden belirlenmelidir. Esnek Gerçek-Zamanlı sistemlerde ise, daha esnek zaman kısıtları vardır; bazı zaman kısıtlarına uyulmayabilir. Oluşan hatalar nedeni ile zamanlama kısıtlamaları uygulamanın belirli bir periyodunda sağlanamazsa performans düşer fakat sistem kullanım dışı kalmaz. Kullanılan veriler halen geçerli olabilir (Livani vd., 1998) (Ercek vd., 2005).

CAN, özellikle katı gerçek-zamanlı haberleşme gereksinimleri olduğunda, sıklıkla kullanılan haberleşme protokollerinden biridir (Shokry, vd., 2009). CAN ağındaki mesaj zamanlamasının amacı, iletim hattı (bus) üzerindeki mesaj iletiminin nasıl kontrol edileceğini belirlemektir. Bunun için mesajlara öncelik verilir. İletim önceliği, ağ modülünden çıkmakta olan trafiğin hangi sırada iletileceğini belirler. İletim önceliği olan mesaj, iletim önceliği olmayandan daha önce iletilir. İletim öncelikleri işlemlerin kritik ve önemli görevler içerip içermediklerine göre ele alınır.

Güvenilir zamanlama davranışları, gerçek-zaman gereksinimleri olan kontrol sistemleri için çok önemlidir. Sistem geliştiricileri, uygun zamanlama mekanizmaları kullanarak, tasarım aşamasında zamanlama davranışlarını planlayabilirler (Michta, 2005). Gerçek-zamanlı mesajların zamanında tamamlanabilmesi için, özellikle planlanması gerekir (Zuberi ve Shin, 2000).

Zamanlama algoritmalarının amaçları şunlardır:

- İsteklere hızlı bir şekilde cevap verilmelidir.
- Kullanıcının beklentilerini karşılamalıdır.
- Son teslim süresine (deadline) riayet edilmelidir.
- Veri kaybından sakınılmalıdır.
- Sürecin (işlemin) sunumu ve sonlanması arasındaki süre minimum olmalıdır.
- Üretilen iş maksimum olmalıdır.

Zamanlamada görevler, periyodik ya da aperiyodik olabilirler. Periyodik görevler, belli sabit zaman aralıklarında kendini tekrar ederken, aperiyodik görevler belli bir zaman aralığı olmadan, herhangi bir anda gerçekleştirilebilir. Aperiyodik görevler, sistem tamamlanıncaya kadar sadece bir defa aktive edilirken, periyodik görevler defalarca gerçekleştirilir ve art arda gelen iki aktivasyon arasındaki gecikme, periyot değerine eşit sabit bir değerdir.

Zamanlama algoritmaları, kesintili (Preemptive) ve kesintisiz (Non-Preemptive) olmak üzere ikiye ayrılır (Rouhifar ve Ravanmehr, 2015). Kesintili algoritmalarda, düşük öncelikli bir mesajın iletimi esnasında yüksek öncelikli bir mesaj gelirse, düşük öncelikli mesajın iletimi askıya alınır. Yüksek öncelikli mesajın iletimi bittikten sonra, düşük öncelikli mesaj iletime kaldığı yerden devam eder. Kesintisiz (sonsuz önceliksiz) algoritmalarda ise yüksek öncelikli bir mesaj gelse bile iletim devam eder. İşlem kendi isteği ile sonlanana kadar kesintisiz çalışır.

Haberleşme sistemlerinde yaygın olarak öncelik-temelli zamanlama benimsenmiştir. Öncelik-temelli zamanlama algoritması, statik ya da dinamik olabilir. Eğer mesaj öncelikleri değiştirilebiliyorsa dinamik, değiştirilemiyorsa statik (sabit) zamanlama olarak adlandırılır. Dinamik zamanlamada, hangi mesajın öncelikli olarak iletileceği kararı, çalışma esnasında verilir (Chen ve Yen, 2012). Statik zamanlamada ise, önceliklendirme kararları sistem çalışmaya başlamadan önce verilmiş olmalıdır. Statik öncelikli zamanlama algoritmalarının tersine dinamik zamanlama algoritmaları, mesajlara her seferinde farklı öncelikleri atar. Statik zamanlamanın avantajı, uygulanmasının kolay olmasıdır. Statik önceliklendirme temelli algoritma, Tek Düze Hız (Rate Monotonic – RM) ve Tek Düze Zaman Sınırı (Deadline Monotonic – DM) olmak üzere ikiye ayrılır.

Makalenin ikinci kısmında bilimsel yazın taraması yapılarak, bu konuda yapılan çalışmalardan bahsedilmiştir. Üçüncü kısımda CAN ağları, dördüncü kısımda RM algoritması ve beşinci kısımda SocketCAN uygulaması hakkında bilgiler verilmiştir. Altıncı bölümde tasarlanan sistem modeli anlatılmıştır. Son bölümde sonuçlar sunulmuştur.

Bu çalışmanın önemi, CAN ağında mesajları statik olarak önceliklendirme işleminin yapılarak, optimum mesaj üretiminin ve iletiminin sağlanmasıdır. Statik zamanlama için RM algoritması kullanılmıştır.

2. Bilimsel Yazın Taraması

Bu çalışmada kullanılan Tek Düze Hız (Rate Monotonic - RM) zamanlama algoritması ilk olarak Liu ve Layland (1973) tarafından çalışılmıştır. Yazarlar, bu algoritmanın analizini bir katı gerçek-zaman ortamında gerçekleştirmiştir.

Tindell vd. (1994) yaptıkları çalışmada RM algoritmasına dayalı bir CAN analizi geliştirmiş ve bir CAN veri yolunda iletilmekte olan mesajların yanıt süresinin nasıl bulabileceğini göstermişlerdir.

Livani vd. (1998) bir CAN ağında katı ve esnek gerçek zamanlı haberleşme sistemleri için sabit ve dinamik öncelikli zamanlamanın bir kombinasyonunu çalışmışlardır.

Fohler vd., (2001) gerçek-zamanlı zamanlama teknikleri üzerine yaptıkları araştırmalarda EDF ve çevrim dışı algoritmanın örneklerini vermişlerdir.

Dobrin ve Fohler (2001), yaptıkları çalışmada CAN ağında çevrimdışı mesaj zamanlaması yöntemi sunmuşlardır. Sabit öncelikli bir dizi periyodik mesaj türeterek, analiz yapmışlardır.

Oliveira vd. (2003), RM algoritması kullanarak bir CAN bus üzerinde zamanlama uygulaması gerçekleştirerek, katı gerçek zamanlı sistemlerde hata tolerans kapasitesini incelemişlerdir.

Aysan vd. (2010), farklı kriterlere sahip mesajların CAN ağında zamanlamasının uygulanması için bir metodoloji önermişlerdir. Önerilen yaklaşım, kritik mesajlar için hata toleransı olan optimal mesaj önceliklerinin çevrim dışı türetilmesini içermektedir.

Shinde ve Biday (2017), çalışmalarında gerçek zamanlı statik ve dinamik zamanlama algoritmalarının karşılaştırmalarını yapmışlardır.

3. Kontrol Alan Ağı (CAN)

CAN, yüksek performanslı ve yüksek güvenirliği olan gerçek zamanlı veri haberleşmesi için uygun bir protokoldür. CAN, başlangıçta otomotiv endüstrisinde kullanılmak üzere tasarlanmış olsa da, günümüzde birçok endüstriyel kontrol uygulamalarında yaygın olarak tercih edilmektedir. Bir CAN ağı, iki ya da daha fazla düğümün, CAN protokolü ile bir CAN bus'a (veriyolu) bağlanmasıyla oluşur. Ağ performansını etkileyen anahtar faktör, CAN protokolünün doğasından kaynaklanan mesaj öncelik yapısıdır. CAN mesajlar, CAN veri yoluna önceliklerine bağlı olarak erişirler. CAN bus, hızı 10 kbps'den 1Mbps'e kadar çalışmak üzere dizayn edilmiş bir broadcast (yayın) veri yoludur.

CAN protokolü, çarpışma denetimi için CSMA (Carrier Sense Multiple Access / Çarpışma Denetimi ile Taşıyıcı Algılamalı Çoklu Erişim) erişim mekanizması esasına göre çalışan CSMA/CR (Carrier Sense Multiple Access with Collision Resolution / Çarpışma Çözümü ile Taşıyıcı Algılamalı Çoklu Erişim) kullanır (Anwar ve Khan, 2007). İletime geçmek isteyen düğüm, önce ağın boş olup olmadığını kontrol eder. Eğer ağda iletim yoksa düğüm mesajı göndermeye başlar. Veri iletim yolu, bağlı olan tüm düğümlerin veri aktarımına açık olduğu için aynı anda farklı düğümler iletime geçmeye çalışırsa çarpışma meydana gelir. Çarpışma sonrası düğümler, sıkışma (jam) sinyali göndererek, geri çekilirler. Çarpışma durumunda tüm veriler bozulur ve yeniden aktarılması gerekir. Bu durum ağ performansını olumsuz yönce etkileyerek fazladan ağ trafiği ve gecikme oluşmasına neden olur. Bu nedenle veri gönderen bir düğümün aktarım sonrası hattı dinlemesi ve olası çarpışmaların farkına varması gerekir.

CSMA/CR erişim modelinde, her bir mesaj, tek ve benzersiz bir öncelik ile karakterize edilir. Aynı anda iki ya da daha fazla düğüm iletime geçerse, çarpışma oluşmasını önlemek için, yüksek öncelikli mesaj iletime devam ederken, diğeri yol boşalana kadar bekler. Bu sayede bir kontrol ağında var olan düğüm sayısı kadar farklı öncelik değeri verilebilir.

3.1. CAN Çerçeveleri

Bir CAN bus üzerinde mesajlar, çerçevelerle iletilir. CAN ağı iki farklı çerçeve (frame) formatında konfigüre edilebilir:

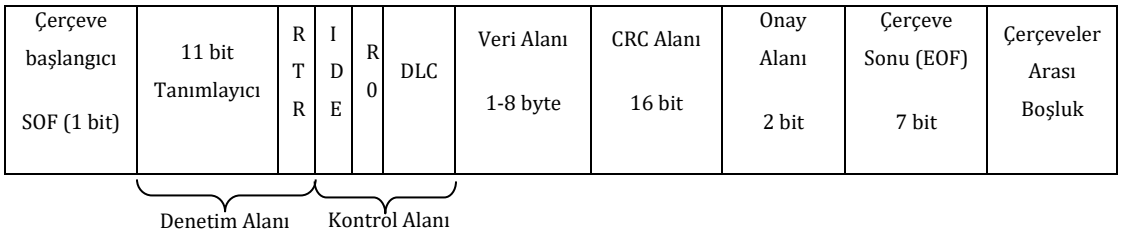
1. Standart ya da temel çerçeve formatı (CAN 2.0 A)
2. Genişletilmiş çerçeve formatı (CAN 2.0 B)

İki çerçeve formatı arasındaki tek fark, standart CAN çerçevesi 11 bitlik tanımlayıcıyı desteklerken, genişletilmiş CAN çerçevesinin 29 (11+18) bitlik tanımlayıcıyı desteklemesidir (Bosch, 1991).

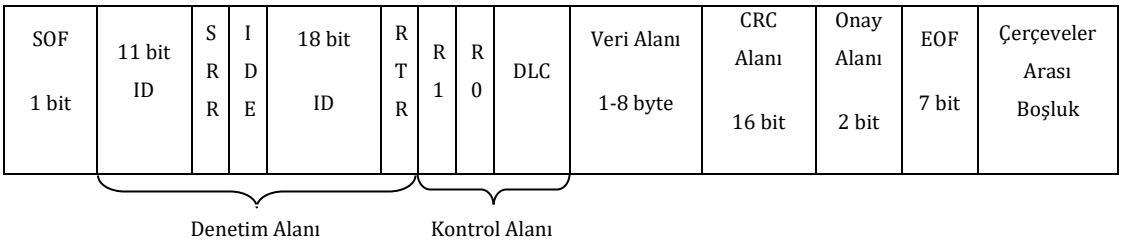
CAN çerçevelerinin içinde gönderici ya da alıcı adresleri yoktur. Onun yerine mesaj içeriğini tanımlayan bir tanımlayıcı alan kullanırlar. Alıcı istasyon bu tanımlayıcı alana bakarak mesajın kendisine gelip gelmediğini anlar. Mesajın kimden geldiğinin önemi yoktur. 11 bitlik tanımlayıcı ile 2048 farklı istasyon adreslenebilirken, 29 bitlik tanımlayıcı ile 536 milyondan fazla farklı istasyon adreslenebilmektedir. Standart çerçeve formatı daha yaygın olarak kullanılırken, genişletilmiş çerçeve formatı endüstride kamyon ve tır gibi araçların elektronik aksamlarının haberleşmesinde ve CAN protokolü tabanlı yüksek seviyeli protokollerin geliştirilmesinde kullanılmaktadır.

Bu çalışmada standart CAN çerçeve formatı kullanılmıştır. Standart ve Genişletilmiş CAN çerçeveleri; çerçeve başlangıcı (SOF), denetim, kontrol, veri, CRC, Onay (ACK), çerçeve sonu (EOF) ve çerçeveler arası boşluk (IFS) olmak üzere 8 alandan

oluşmaktadır. Şekil 1'de Standart CAN çerçeve (CAN 2.0 A) alanları, Şekil 2'de ise Genişletilmiş CAN çerçeve (CAN 2.0 B) alanları görülmektedir (Farsi vd., 1999).



Şekil 1. Standart CAN çerçeve (CAN 2.0 A) alanları



Şekil 2. Genişletilmiş CAN çerçeve (CAN 2.0 B) alanları

3.2. CAN Haberleşmesi

CAN protokolü, adres bazlı bir protokol olmayıp, mesaj bazlı bir protokoldür. Gönderilen tüm mesajlar, ağda bulunan diğer tüm ağ düğümlerine broadcast şeklinde yayılır (Bosch, 1991), (Davis vd., 2007).

CAN, OSI modelinin ikinci katmanı olan Veri Bağı katmanında çalışır. Bu katmanda kullanılan veri yapısı çerçeve olduğu için, CAN ağında veriler çerçeveler ile gönderilir ve alınır. Gönderilen veride, mesajın kaynak ve hedef adreslerini yoktur. Onun yerine, her bir mesaj, ağ üzerinde tek ve benzersiz olan bir tanımlayıcı (ID) tarafından etiketlenir. Bu tanımlayıcılar mesaj içeriğini tanımladıkları gibi, aynı zamanda mesajların önceliklerini de gösterirler. CAN bus için yarışan mesajlar bu önceliklere göre haberleşmeye başlarlar.

CAN bus üzerinde haberleşmek isteyen düğümlerin veri yolunun kullanımı konusunda anlaşmazlığa düşmemeleri için bit hakemliği kavramı geliştirilmiştir. Hakemlik mekanizmasında, mesajlar gönderilirken, öncelik sıralaması yapılmakta ve böylece, yüksek önceliğe sahip olan mesaj gecikme olmadan gönderilebilmektedir. Bit seviyesinde hakemlik işleminin gerçekleşebilmesi için lojik durumların dominant (baskın) ve çekinik (resesif) olarak tanımlanması gerekir. CAN'da, 0 biti dominant, 1 biti ise çekinik olarak tanımlanır. Bus üzerinde aynı anda farklı düğümlerden 0 ve 1 biti gelmesi durumunda 0 biti, 1 bitine karşı baskın gelir. Yani, bit dominant ise, her zaman hakemlik yarışını kazanır (Tindell vd., 1994).

İletime geçmek isteyen bir düğüm, iletim hattını sürekli olarak dinler. Eğer kanalı boş bulursa, bir SOF (Start of Frame) biti yayınlayarak hakemlik evresine başlar. Bu noktada, gönderilecek mesajı olan her düğüm, mesajın tanımlayıcı bitlerini kanala iletmek için yarışabilir. Tanımlayıcı, iletilecek mesajın ilk kısmıdır (Natale, 2000).

Tanımlayıcı bitleri arasındaki çarpışmaların bir mantıksal "VE" işlemi ile çözüldüğü farz edilir. Eğer bir düğüm, bir değişiklik olmadan bus üzerindeki öncelik bitlerini okuyorsa, hakemliği kazanır ve iletim haklarını alır. Eğer bu bitlerden biri bus'tan okunurken daha yüksek öncelikli bir mesaj bus'a girerse, bu durumda mesaj geri çekilir.

Gönderilen mesaj, iletim hattı üzerindeki her düğüme ulaşır. Sadece bu mesajla ilişkisi olan düğümler mesajı okuyup işler. Eğer yol boşaldığında birden fazla düğüm mesaj göndermeye başlarsa düşük ID'li mesaj yolu ele geçirir ve diğer düğümler aradan çekilerek tekrar göndermek üzere yolun boşalmasını beklerler.

4. Rate Monotonic (RM) Zamanlama Algoritması

Tek Düze Hız (Rate Monotonic - RM) algoritması, gerçek zamanlı uygulamalarda en çok tercih edilen algoritmalardan biridir. RM zamanlama algoritması ilk olarak Liu ve Layland (1973) tarafından çalışılmıştır.

Yazarlar, bu ünlü algoritmanın analizini bir katı gerçek-zaman ortamında gerçekleştirmiştir. RM'nin tüm sabit öncelikli şemalar arasında optimal olduğunu, yani bir görev kümesinin RM tarafından zamanlanamazsa, başka herhangi bir sabit öncelikli görev tarafından zamanlanamayacağını kanıtlamıştır (Bini ve Buttazzo, 2004). Çeşitli kısıtlamaları olmasına rağmen, uygulaması kolay olduğu için popüler bir algoritmadır. Sabit ve bilinen öncelik düzeni olan sabit görevler kümesi için, tüm görevlerin son teslim sürelerini karşılaması için sistem kullanımını garanti eder (Michta, 2005).

Statik öncelikli zamanlama algoritmalarında, mesajların öncelikleri, yayınlanmadan önce sadece bir kez atanır ve sistem çalıştığı sürece değiştirilemez. Mesajlar periyodik olmalı ve sonraki mesaj gönderilmeden önce iletimini bitirmeli yani hedefe ulaşmalıdır. Bir başka deyişle, mesajın son teslim süresi, periyoduna eşit olmalıdır (D=T). Bir görevin önceliği, bir görevin yürütme sıklığı ile ilgilidir, böylece en yüksek frekans görevleri en yüksek önceliğe sahip olur. Bu, en yüksek öncelikli görevin en kısa süreye sahip olduğu anlamına gelir. (Mutka, 1994). Tüm mesajlar, periyotlarına göre zamanlanmalıdır (Harkut vd., 2014). Kısa periyoda sahip olan mesaja en yüksek öncelik verilir.

Sabit zamanlama algoritmalarının avantajları, kanal bandgenişliğinden verimli faydalanması, ek yüklerin (overhead) küçük olması, esnek bir yapıya sahip olması ve uygulamasının kolay olmasıdır (Dobrin ve Fohler, 2001). Dezavantajı ise nispeten yavaş olması sebebiyle, sistemin aşırı yüklenmesidir. Bununla birlikte, dinamik öncelikli zamanlamayla kıyaslandığında, yüksek bandgenişliği kullanamaz (Natale, 2000).

5. SocketCAN

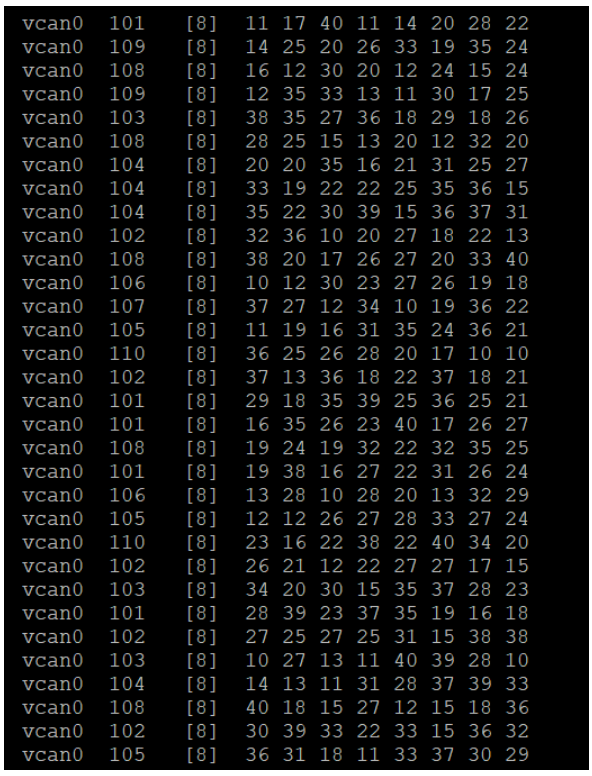
Socketcan, Volkswagen Research tarafından geliştirilmiş, Linux işletim sistemi altyapısını kullanan açık kaynak kodlu bir CAN sürücüsüdür (Sojka vd., 2010). SocketCAN'ın temel amacı, Linux ağ katmanı üzerine kurulan kullanıcı alanı uygulamalarına bir soket arabirimi sağlamaktır. Yaygın olarak bilinen TCP / IP ve Ethernet ağlarının aksine, CAN veri yolu Ethernet gibi hiçbir MAC katmanı adresleme özelliğine sahip olmayan bir broadcast ortamıdır. Linux çekirdeği, tam özellikli bir CAN ağı alt sistemi içerir. Uygulamalara birkaç farklı arayüz üzerinden erişilebilir. CAN-utils paketi ile birlikte, CAN ağları ile çalışmak için çok yönlü ve kullanıcı dostudur (Sojka vd., 2014). Birden fazla uygulamanın aynı anda bir CAN aygıtına erişmesine izin verir. Ayrıca, tek bir uygulama birden fazla CAN ağına paralel olarak erişebilir. SocketCAN'ın temel bileşenleri, farklı CAN denetleyicileri için ağ aygıt sürücüleri ve CAN protokol ailesinin uygulanmasıdır.

Socketcan, Berkeley soket uygulama programlama arayüzünü (API) kulanır. Programlamada API, yazılım uygulamaları inşa etmek için takip edilen rutinler, uygulanan protokoller ve kullanılan araçlar bütünüdür. CAN denetleyici donanımı için cihaz sürücüleri, kendilerini bir ağ cihazı gibi Linux ağ katmanı ile kaydederler. Böylece, denetleyiciden gelen CAN çerçeveleri ağ katmanından ve CAN protokol modülü üzerinden geçebilir. Aynı zamanda, protokol aile modülü iletim protokolü modüllerini kaydetmek için bir API sağlar. Bu sayede herhangi bir sayıdaki iletim protokolleri dinamik olarak yüklenebilir ya da kaldırılabilir. Aslında, CAN çekirdek modülü tek başına herhangi bir protokol sağlamaz ve en az bir ek protokol modülü yüklemeden kullanılamaz. Aynı anda, farklı ya da aynı protokol modülünde birden fazla soket açılabilir. Bu soketler, farklı ya da aynı CAN ID'sine sahip çerçeveleri dinleyebilir ve gönderebilirler. Aynı CAN ID'si olan çerçeveler için aynı arayüzü dinleyen çeşitli soketlere, hepsi aynı alınan CAN çerçeve eşleşmeleri iletilir (Github, 2017).

SocketCan, sanal CAN (VCAN) cihazları oluşturarak ağ arayüzlerinin kullanılmasını sağlayan bir yazılımdır. Böylece CAN protokolleri herhangi bir donanıma gereksinim duymadan da çalıştırılabilmektedir. Sanal CAN cihazları genellikle vcan0, vcan1, vcan2… şeklinde adlandırılır (Github, 2010).

6. Tasarlanan Sistem Modeli

Bu çalışmada, kesintisiz zamanlama ile RM algoritması kullanılmıştır. Veri iletim hızı 125 kbps olarak ayarlanmıştır. CAN ID'ler, hakemlik için kullanıldığından birbirlerinden farklı olarak seçilmiştir. Tasarlanan ağ modelinde toplamda 10 tane farklı ID değeri olan 100 adet mesaj üretilmiştir. Her bir mesaja 101 ile 110 arasında onaltılık (hex) değerler halinde farklı ID'ler verilmiştir. Üretilen mesajlardaki veriler 8 byte uzunluğundadır ve random olarak belirlenmiştir. Şekil 3'de tasarladığımız ağ modelinde üretilen mesajların önceliklendirme yapılmadan önceki akış trafiğinin bir kısmı görülmektedir. Burada birinci sütun CAN bus hattını, ikinci sütun mesajın ID değerini, üçüncü sütun verinin boyutunu ve dördüncü sütun ise iletilen mesajı belirtmektedir.



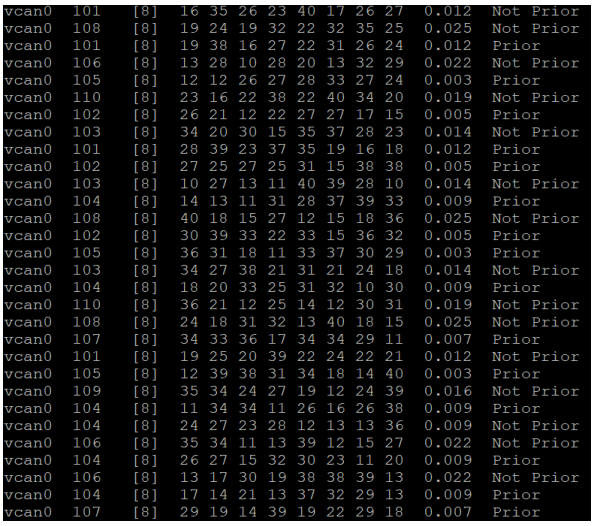
Şekil 3. Önceliklendirme yapılmamış mesaj trafiğinin bir kısmı

Daha sonra, mesajlara öncelik atamaları yapılmıştır. Tasarlanan ağ modelinde, mesajların son teslim süreleri ile periyodları eşittir. Her bir mesajın önceliği sabittir ve sistem çalışmadan önce ayarlanmıştır. Mesajların son teslim süreleri 3 ms ile 25 ms arasında seçilmiştir. Son teslim süresi en düşük olan mesaj en yüksek öncelik değerini alırken, son teslim süresi en yüksek olan mesaj ise en az önceliklidir. Mesajların ID değerleri, öncelik durumları ve son teslim süreleri, Tablo 1'de verilmiştir.

Tablo 1. Mesajların ID değerleri, öncelik durumları ve son teslim süreleri

ID	Öncelik Durumu	Son Teslim Süresi (Deadline)
105	1.	3 ms
102	2.	5 ms
107	3.	7 ms
104	4.	9 ms
101	5.	12 ms
103	6.	14 ms
109	7.	16 ms
110	8.	19 ms
106	9.	22 ms
108	10.	25 ms

İletilmek üzere CAN bus hattına gelen mesajın son teslim süresine bakılarak hakemlik mekanizması çalışır. Önceliklendirme işlemi yapıldıktan sonra oluşan akış trafiğinin bir kısmı Şekil 4'de görülmektedir. Burada birinci sütun CAN bus hattını, ikinci sütun mesajın ID değerini, üçüncü sütun verinin boyutunu, dördüncü sütun iletilen mesajı, beşinci sütun mesajın son teslim süresini (deadline) ve altıncı sütun ise öncelik durumunu göstermektedir. "Prior" ifadesi mesajın öncelik aldığını, "Not Prior" ifadesi ise öncelik almadığını belirtmek için kullanılmıştır.



Şekil 4. Önceliklendirme yapıldıktan sonra oluşan mesaj trafiğinin bir kısmı

5. Sonuç ve Tartışma

Gerçek-zaman performansı; otomobiller, medikal cihazlar, savunma sistemleri, nükleer sistemler ve uçak uçuş kontrolü gibi çok kritik sistemlerde en önemli problemlerden biri haline gelmiştir. Bu kritik sistemler için haberleşme ağının, katı gerçek-zaman davranışı sergilemesi beklenir. Her bir mesajın zaman sınırlamalarını karşılamak için zamanlama yapmak, bu tip güvenilir sistemler için oldukça önemlidir. Zamanlama, önem derecesine göre iletilecek olan mesajların önceliklendirilmesi ile yapılır.

CAN, zorlu koşullarda yüksek güvenilirliğe sahip çok sayıda kısa mesaj gerektiren endüstriyel uygulamalar için ideal bir gerçek zamanlı haberleşme protokolüdür. Bu çalışmada, Linux tabanlı SocketCAN yazılımı aracılığıyla bir CAN ağı tasarlanmış ve iletilecek mesajların hangi öncelikte olacağı sistem çalışmaya başlamadan önce belirlenmiştir. Statik olarak önceliklendirme yapılan bu mesajların iletimi, son teslim süreleri içinde herhangi bir bozulma olmadan başarıyla gerçekleştirilmiştir. CAN ağı tasarlanırken, gerçek zamanlı mesaj trafiğini desteklemesi ve uygulaması temel alınmıştır. Tasarım, yüksek öncelikli mesajların son teslim sürelerini karşıladığını ve daha düşük öncelikli mesajların yüksek öncelikli mesajları engellemediğini

kanıtlamıştır. Gelecekteki çalışmalarda, farklı senaryolar için yeni ağ modelleri tasarlanarak, statik öncelik temelli SocketCAN uygulaması pek çok endüstriyel ağda etkili olarak kullanılabilir.

Conflict of Interest / Çıkar Çatışması

Yazarlar tarafından herhangi bir çıkar çatışması beyan edilmemiştir.
No conflict of interest was declared by the authors.

Kaynaklar

Anwar, K., Khan, Z.A., 2007. Dynamic Priority Based Message Scheduling on Controller Area Network. Proceedings of International Conference on Electrical Engineering (ICEE), 1–6 April.

Aysan, H., Thekkilakattil, A., Dobrin, R., Punnekkat, S., 2010. Efficient fault tolerant scheduling on controller area network (CAN). 15th International Conference on Emerging Technologies and Factory Automation, 13-16 September.

Bini, E., Buttazzo, G.C., 2004. Schedulability analysis of periodic fixed priority systems. IEEE Transactions on Computers, 53(11), 1462-1473.

Bosch, R., CAN Specification Version 2.0, Stuttgart, Germany, 1991.

Chen, M., Yen, H.W., 2012. An Online RBF Network Approach for Adaptive Message Scheduling on Controller Area Networks. Journal of Information Science and Engineering, 28, 503-519.

Davis, R., Burns, A., Bril, R.J., Lukkien, J.J., 2007. Controller Area Network (CAN) Schedulability Analysis: Refuted Revisited and Revised. Real-Time Systems, 35(3), 239-272.

Dobrin, R., Fohler, G., 2001. Implementing off-line message scheduling on controller area network (CAN). Proceedings of Emerging Technologies and Factory Automation. Antibes-Juan Les Pins, 241-245.

Ercek, E., Erdoğdu, S., Uluat, M. F., 2005. Gerçek Zamanlı Sistemlerde UML Uygulamaları. EMO II. İletişim Teknolojileri Ulusal Sempozyumu, Adana.

Farsi, M., Ratcliff, K., Barbosa, M, 1999. An overview of controller area network, Computing & Control Engineering Journal, 10(3), 113-120.

Fohler, G., Lenvall, T., Buttazzo, L.G., 2001. Improved Handling of Soft Aperiodic Tasks in Offline Scheduled Real-Time Systems using Total Bandwidth Server, 8th IEEE International Conference on Emerging Technologies and Factory Automation, 15-18 October, France, 151-157.

Github, 2010. https://github.com/rhyttr/SocketCAN/commit/ db198f35c3e9ab17c62 ed63d238f1c4afa9acb19.

Github, 2017. https://github.com/rhyttr/SocketCAN.

Harkut, D.G., Ali, M.S., Lohiya, P., 2014. Real-time scheduling algorithms for wireless sensor network. Circuits and Systems: An International Journal CSIJ, 1(1), 11-18.

Liu, C.L., Layland, J.W. 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of ACM, 20(1), 46–61.

Livani, M.A., Kaiser, J., Jia, W.J., 1998. Scheduling Hard and Soft Real-Time Communication in The Controller Area Network (CAN). Proceedings of the 23rd IFAC Workshop on Real-Time Programming, June, 13-18, China.

Michta, E., 2005. Scheduling Systems. Sydenham, P.H., Thorn, R., (Ed.), Handbook of Measuring System Design, John Wiley and Sons, 1648p.

Mutka, M.W., 1994. Using Rate Monotonic Scheduling Technology for Real-Time Communications in a Wormhole Network. Proc. Second Workshop on Parallel and Distributed Real-Time Systems, Cancun, Mexico, 194-199.

Natale, M.D,. 2000. Scheduling the CAN Bus with Earliest Deadline Techniques. In Proceedings of the 21st IEEE Real-Time Systems Symposium, December 2000, 259–268.

Oliveira, M.P., Fernandes, A.O., Campos, S.V.A., Zuquim, A.L.A.P., Mata, J.M., 2003. Guaranteeing Fault Tolerance through Scheduling on a CAN bus. The International CAN Conference (ICC2003), 15-22.

Rouhifar, M., Ravanmehr, R., 2015. A Survey on Scheduling Approaches for Hard Real-Time Systems. International Journal of Computer Applications, 131(17), 41-48.

Shinde, V., Biday, S.C., 2017. Comparison of Real Time Task Scheduling Algorithms. International Journal of Computer Applications, 158(6), 37-41.

Shokry, H., Shedeed, M., Hammad, S., Shalan, M., Wahdan, A., 2009. Hardware EDF scheduler implementation on controller area network controller. In Proceedings of 4th International IEEE Design and Test Workshop (IDT) 2009, Riyadh, Saudi Arabia.

Sojka, M., Píša, P., Petera, M., Špinka, O., Hanzálek, Z., 2010. A Comparison of Linux CAN Drivers and their Applications. 5th IEEE International Symposium on Industrial Embedded Systems (SIES), 7-9 July, 18-27.

Sojka, M., Píša, P., Hanzálek, Z., 2014. Performance evaluation of Linux CAN-related system calls. 10th IEEE Workshop on Factory Communication Systems (WFCS 2014), 5-7 May, Toulouse, France, 1-8.

Tindell K., Burns A., Wellings A.,. Calculating Controller Area Network (CAN) Message Response Times. University of York, Department of Computre Science, York, England, 1994.

Zuberi, K.M., Shin, K.G., 2000. Design and implementation of efficient message scheduling for controller area network. IEEE Transactions on Computers, 49(2), 182-188.